Statistical report on red wine quality, prediction using R

Student: Salvatore Romano

Prof. Salvatore Ingrassia

University of Catania Master's degree in Data Science for management

Academic year 2021/2022

Contents

1 Data set introduction

The data set is called red wine quality, and it is related to Portuguese "Vinho Verde" wine. The data set that I analyzed has been reviewed by the Professor, and is made up of three data sets:

- *Train data*: about 60% of the units of the original data set
- *Validation data*: about 20% of the units of the original data set
- *Test data*: about 20% of the units of the original data set

The work aims to predict the values of the target variable "quality" (ordinal type, that includes integer numbers ranging from 3 to 8) in the test data set; the quality of the wine is based on various features like *fixed acidity, volatile acidity, citric acid, residual sugar, density, pH* and many more.

A consideration to make is about the target variable quality, that has been reclassified in four values, in fact the values 3 and 4 have been put together in class "4" and the values 7 and 8 have been put together in class "7". So, based on this, the quality variable has 4 levels (4,5,6,7). The train_data data set contains 957 observations and 12 variables (target variable included).

Variable's name	***Type***	***Description***
fixed.acidity	num	most acids involved with wine or fixed or nonvolatile
volatile.acidity	num	the amount of acetic acid in wine, which at too high of levels can lead to an unpleasant, vinegar taste
citric.acid	num	found in small quantities,citric acid can add 'freshness' and flavor to wines
residual.sugar	num	the amount of sugar remaining after fermentation stops
chlorides	num	the amount of salt in the wine
free.sulfur.dioxide	num	the free form of SO2 exist itself and bisulfite ion
total.sulfur.dioxide	num	amount of free and bound forms of S02
density	num	the density of water is close to that of water depending on the percent alcohol and sugar content
ph	num	describes how acidic or basic a wine is
sulphates	num	wine additive which can contribute to sulfur dioxide gas (S02) levels
alcohol	num	the percent alcohol content of the wine
quality (target variable)	ordinal	score from 4 to 7

Table 1: Variable's overview

1.1 Some questions

Definition of some questions that will guide me in this analysis.

1. Which are the variables that present the highest correlation coefficient?
2. Which are the variables that present the lowest correlation coefficient?
3. Which variables contributes the most to target variable quality?
4. How can we predict the target variable quality?

Some other possible questions can emerge during the analysis.

2 Exploratory Data Analysis

The EDA, Exploratory Data Analysis is used to analyze and investigate data sets and summarize their main characteristics. We can divide the EDA in two parts, one that concerns the univariate analysis, and the other one concerns the multivariate analysis.

2.1 Univariate Analysis

Performing the univariate analysis for the training data set, for each variable. I choose to use a summarize univariate analysis. Firstly, with the use of *vis_dat()* we can discover what is inside the train data set; we can also note from Figure 1 that the cells are coloured according to their type.

It can be note that the target variable quality is red.

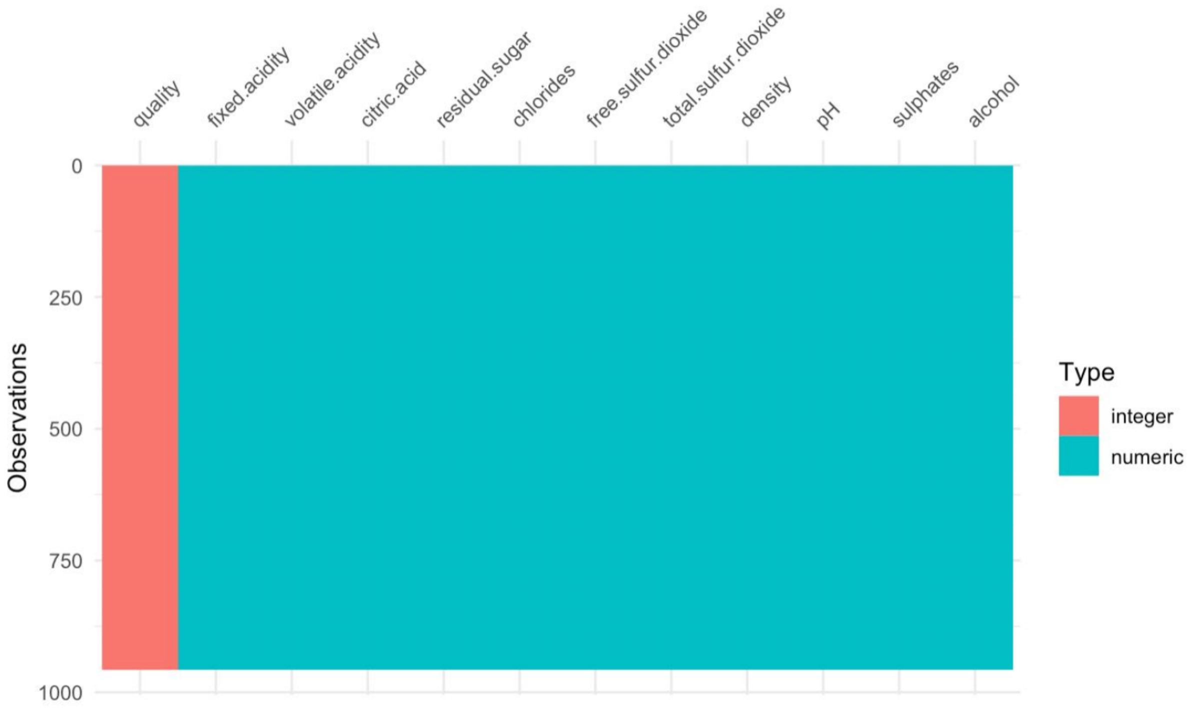
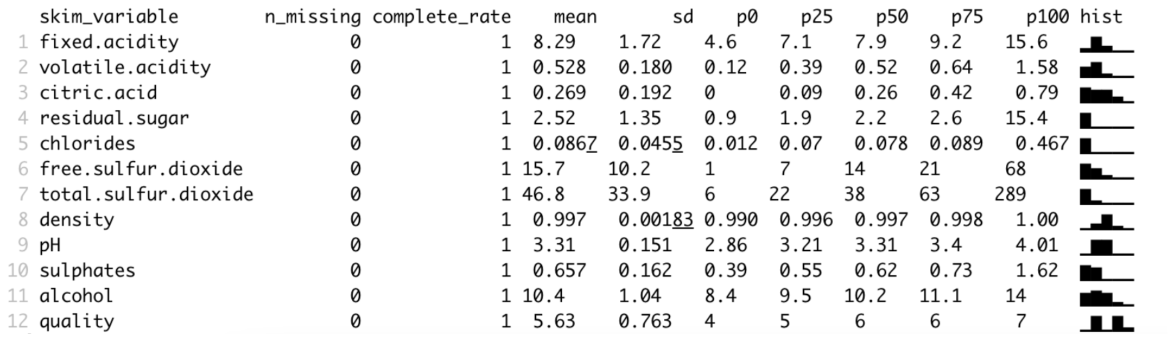


Figure 1: vis_dat output

In order to discover more about the variables, I used *skim()*, a function that allow us to summarize the main statistical features for each variables like the mean, the standard deviation, the minimum and the maximum etc.



```
   skim_variable        n_missing complete_rate    mean       sd     p0     p25    p50    p75    p100 hist
 1 fixed.acidity                0             1  8.29    1.72     4.6    7.1    7.9    9.2    15.6
 2 volatile.acidity             0             1  0.528   0.180    0.12   0.39   0.52   0.64    1.58
 3 citric.acid                  0             1  0.269   0.192    0      0.09   0.26   0.42    0.79
 4 residual.sugar               0             1  2.52    1.35     0.9    1.9    2.2    2.6    15.4
 5 chlorides                    0             1  0.0867  0.0455  0.012  0.07   0.078  0.089   0.467
 6 free.sulfur.dioxide          0             1 15.7    10.2      1      7     14     21      68
 7 total.sulfur.dioxide         0             1 46.8    33.9      6     22     38     63     289
 8 density                      0             1  0.997   0.00183 0.990  0.996  0.997  0.998   1.00
 9 pH                           0             1  3.31    0.151    2.86   3.21   3.31   3.4     4.01
10 sulphates                    0             1  0.657   0.162    0.39   0.55   0.62   0.73    1.62
11 alcohol                      0             1 10.4     1.04     8.4    9.5   10.2   11.1    14
12 quality                      0             1  5.63    0.763    4      5      6      6       7
```

Figure 2: skim output

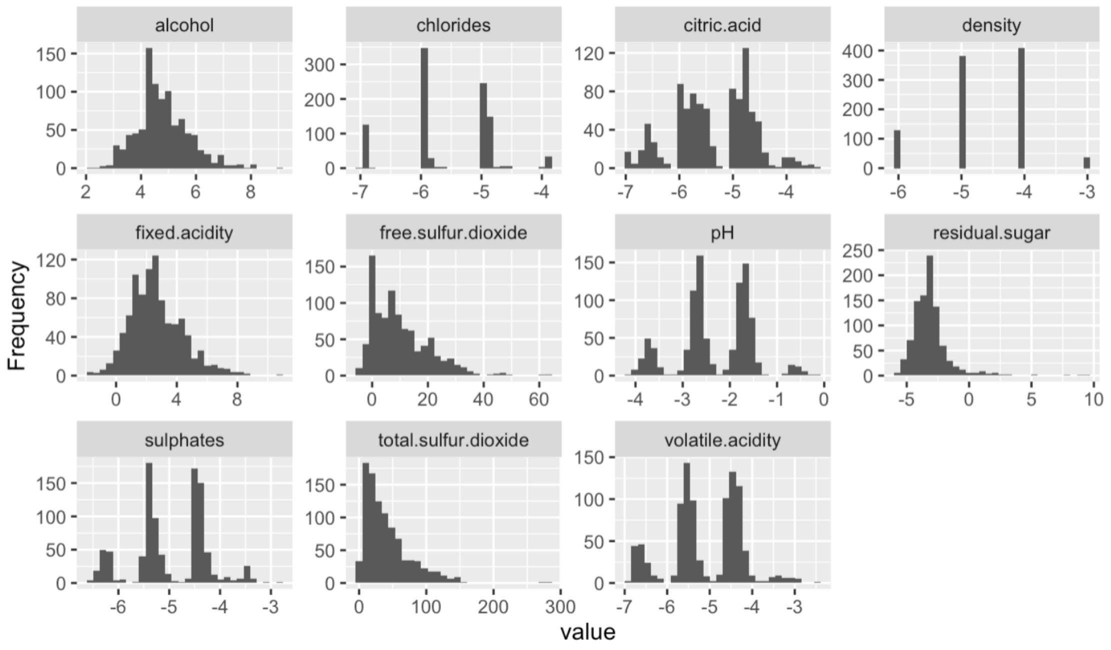


Figure 3: Histograms

In Figure 3 we can see the histograms that show the frequency for each continuous feature. Some variable, according to Figure 3, are leptokurtics, this means they are characterized by long tails.

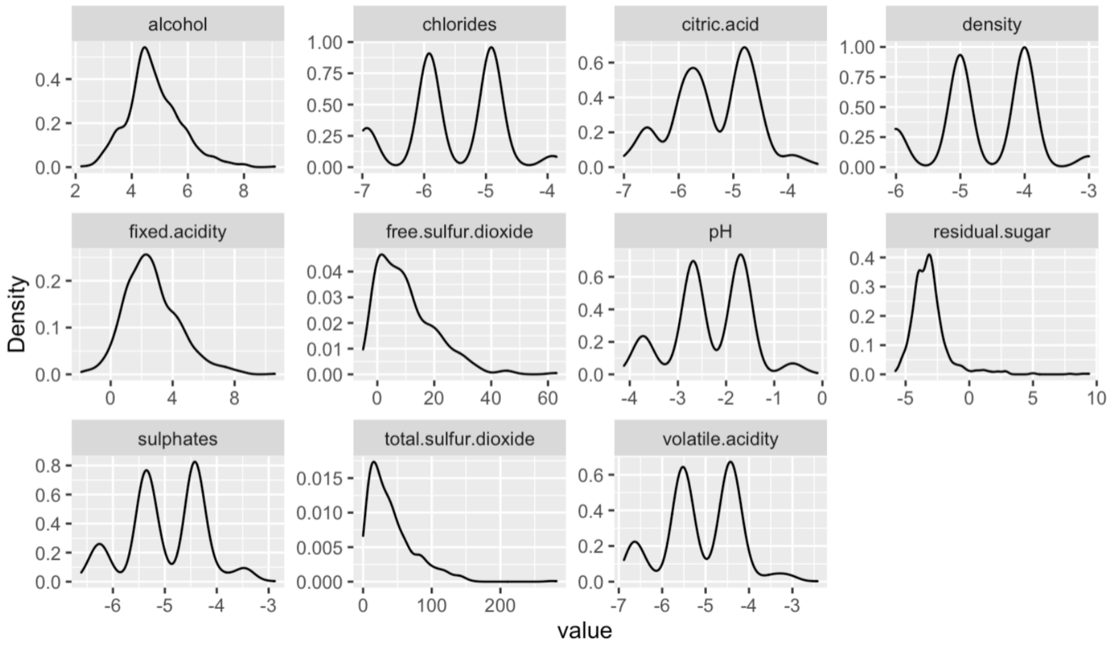


Figure 4: Density lines

In the Figure 4 are displayed the density lines for each continuous feature.

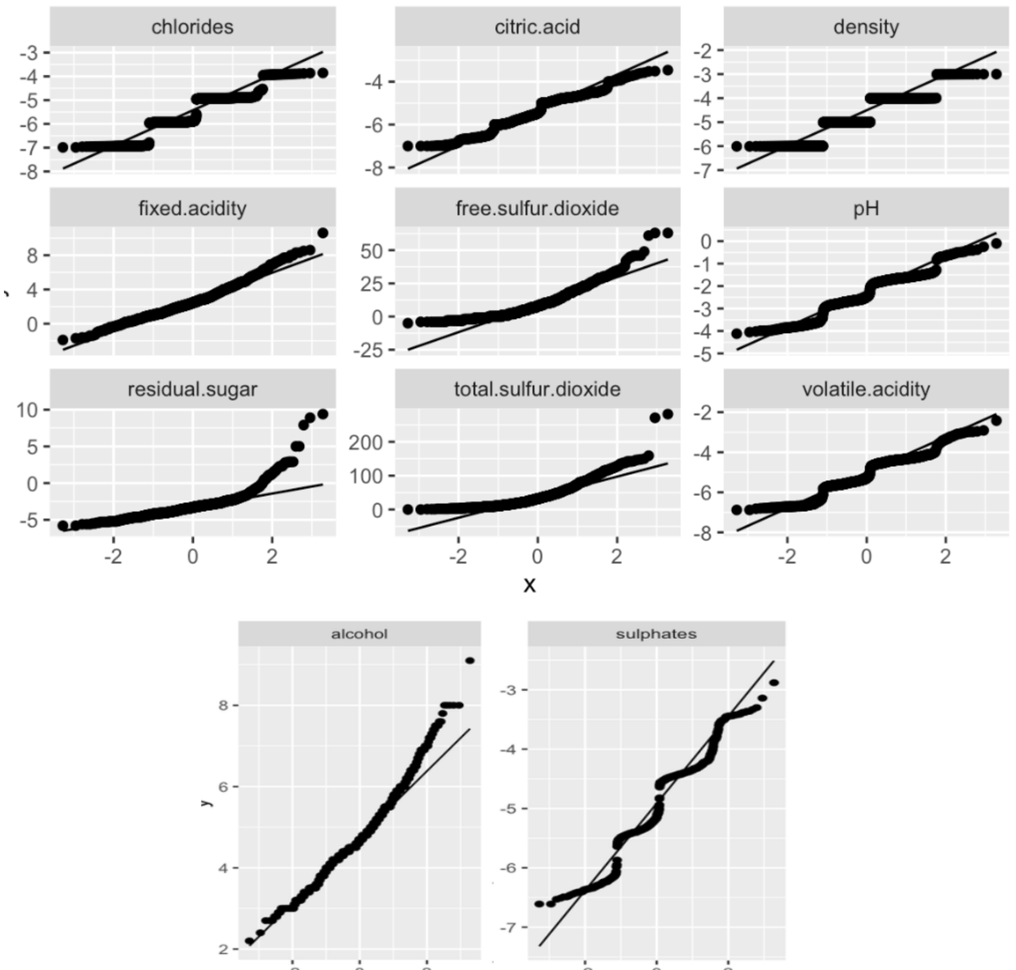


Figure 5: qq-plots

In the Figure 5 are displayed the qq-plots; we can see the presence of a solid lines in all the graphs that represent the expected values. The variable that is nearest to the solid line is *fixed.acidity*.

2.2 Multivariate Analysis

After the univariate analysis, I performed the multivariate analysis that looks at two or more variable at time to explore relationship.

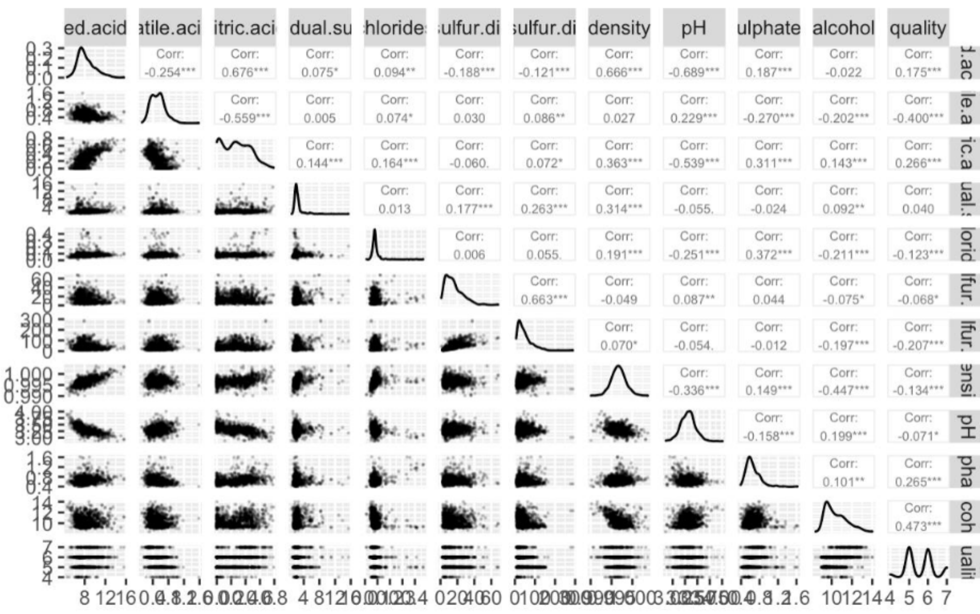


Figure 6: Multivariate conditional scatter plot

The multivariate conditional scatter plot in Figure 6 allows us to see in the upper triangle, the level of correlation among pairwise. As the image reports, there are so many negative values of correlation between pairwise. We can see that the highest correlation value is the pairwise *citric.acid* and *fixed.acidity*, instead the most negative correlated pairwise is *pH* and *fixed.acidity*.

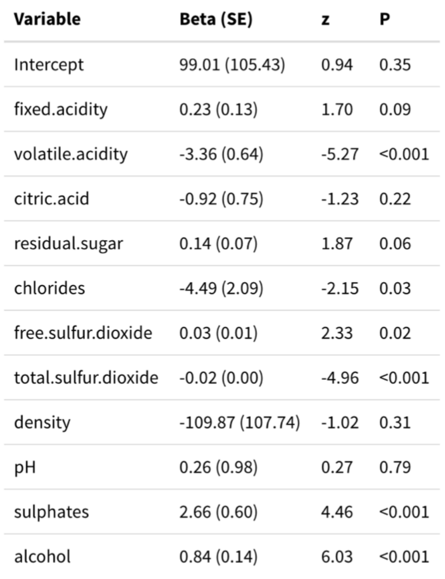
3 Modelling Train Data

The next step of the analysis is related to model the train data set by consider two classes of the target variable quality, more precisely I'll consider 0 the quality with levels 4 or 5, and 1 the quality with levels from 6 to 8. Please note that sometimes we used 0 and 1 and sometimes "low" and "high" due to the nature of the classification model. The analysis is based on the following ML approaches:

- Logistic regression
- Random forest
- Neural network

3.1 Logistic regression

The logistic regression is a classification model used to models the probabilities for classification problems with two possible outcomes. With the use of the *tabglm()* function that belongs to the tab package, we can easily display some relevant information link to the *glm()* output.

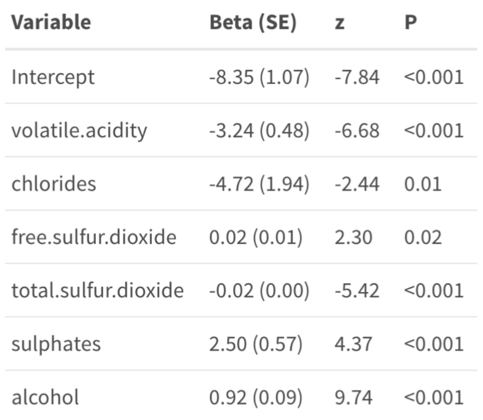


Variable	Beta (SE)	z	P
Intercept	99.01 (105.43)	0.94	0.35
fixed.acidity	0.23 (0.13)	1.70	0.09
volatile.acidity	-3.36 (0.64)	-5.27	<0.001
citric.acid	-0.92 (0.75)	-1.23	0.22
residual.sugar	0.14 (0.07)	1.87	0.06
chlorides	-4.49 (2.09)	-2.15	0.03
free.sulfur.dioxide	0.03 (0.01)	2.33	0.02
total.sulfur.dioxide	-0.02 (0.00)	-4.96	<0.001
density	-109.87 (107.74)	-1.02	0.31
pH	0.26 (0.98)	0.27	0.79
sulphates	2.66 (0.60)	4.46	<0.001
alcohol	0.84 (0.14)	6.03	<0.001

Figure 7: glm() output

As notice by take a look at Figure 7, *Beta(SE)* is the standard error for the unstandardized beta (SE B). This value is like the standard deviation for a mean. The larger the number, the more spread out the points are from the regression line; z is the test-statistic and p is the p-value.

Based on the computation we can update the model, according to the relevant features. In this case, the computation of the logistic regression takes in count only the volatile.acidity, chlorides, free.sulfur.dioxide, total.sulfur.dioxide, sulphates and alcohol.



Variable	Beta (SE)	z	P
Intercept	-8.35 (1.07)	-7.84	<0.001
volatile.acidity	-3.24 (0.48)	-6.68	<0.001
chlorides	-4.72 (1.94)	-2.44	0.01
free.sulfur.dioxide	0.02 (0.01)	2.30	0.02
total.sulfur.dioxide	-0.02 (0.00)	-5.42	<0.001
sulphates	2.50 (0.57)	4.37	<0.001
alcohol	0.92 (0.09)	9.74	<0.001

Figure 8: glm() output relevant features

Going on with evaluation of the logistic regression, by take in count the use of the *contingency matrix* (also known as confusion matrix), a table used to define the performance of a classification model.

	high	***low***	***Sum***
high	390	107	497
low	122	338	460
Sum	512	445	957

Table 2: Contingency matrix on train data set

Based on Table 2, the model's accuracy in the train data set is 76.07% and the error rate is 23.93%. Next, I'll work on the validation data set.

	high	***low***	***Sum***
high	132	43	175
low	41	108	149
Sum	173	151	324

Table 3: Contingency matrix on validation data set

According to Table 3 the accuracy in this case is 74.07% and the error rate is 25.93%.

3.2 Random forest

The next step in modelling train data consists to apply the random forest to the train data set. The random forest is a classification algorithm based on many decision trees which are built and then averages them. According to the computation of the random forest, I selected a number of trees of 500 and a mtry (number of variables randomly sampled as candidates at each split) of 6. The result is an *OOB* (Out Of Bag) estimate of error rate of 19.75%; we can consider the *OOB* a way of validating the random forest model. Note that the random Forest seeks to build trees, so we can have the possibility to get the tree and see what there is inside of it with the function *getTree()*. In the following computation, there is a graph that explains the variable's importance,that allows to discover how much and what are the variables that influence the most the target variable quality.

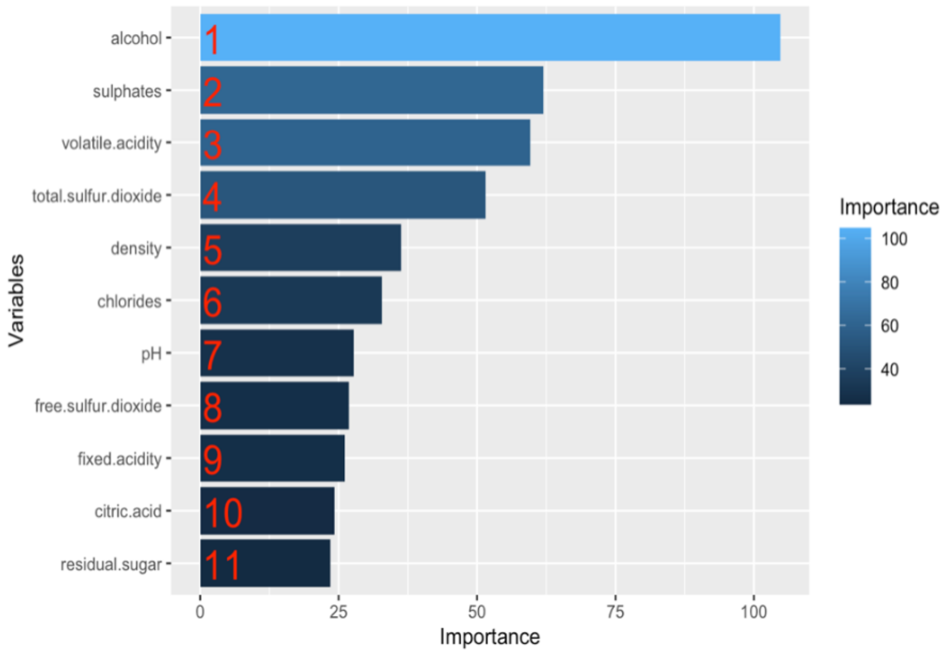


Figure 9: Variables' importance

As we can see, the variables *alcohol* and *sulphates* are the most discriminating factor of wine quality. After the computation of the confusion matrix based on the random forest applied to train data set.

	low	***high***	***Sum***
low	352	93	445
high	89	423	512
Sum	441	516	957

Table 4: Contingency matrix on train data set

Based on the values of Table 4, I computed the accuracy rate that is 81%, and the error rate that is 19%. Next, the computation of the contingency matrix based on the validation data set.

	low	***high***	***Sum***
low	121	30	151
high	29	144	173
Sum	150	174	324

Table 5: Contingency matrix on validation data set

In this case, the accuracy rate improved, is about 81.80%, and the error rate is 18.20%.

3.3 Neural network

The last model that will be able to model the data set is called neural network. The neural network concept is related to machine learning and deep learning; they are made up of node layers, containing an input layer, one or more hidden layers, and an output layer. Remember that neural network approach works much better when the observations are scaled, so the first step of the analysis is scaling the train data set. After the scale, to fit the neural network, there is the building phase, in which I computed the layers and the number of neurons. I built a neural network with 3 blocks. Based on the output layer that contains one neuron that is the target variable quality, I selected a number of units equal to 7. After that, I added some detail to the variable. The structure results, displayed in Table 6, are the following:

Layer(type)	***Output Shape***	***Param#***
dense_3(Dense)	(None,7)	84
dropout_2(Dropout)	(None,7)	0
batch_normalization_2 (BatchNormalization)	(None,7)	28
dense_2(Dense)	(None,7)	56
dropout_1(Dropout)	(None,7)	0
batch_normalization_1 (BatchNormalization)	(None,7)	28
dense_1(Dense)	(None,7)	56
dropout_(Dropout)	(None,7)	0
batch_normalization (BatchNormalization)	(None,7)	28
dense(Dense)	(None,1)	8

Table 6: Neural network structure

The Table 6 describes the neural network's structure and the results; from this I can say that the total parameters are 288, the trainable parameters are 246 and the non-trainable parameters are 42. The output layers, the one that regards the target variable quality, contains 8 parameters. Can we see how the neural network model perform? Yes, we can see it by plotting the training process of our Keras model.

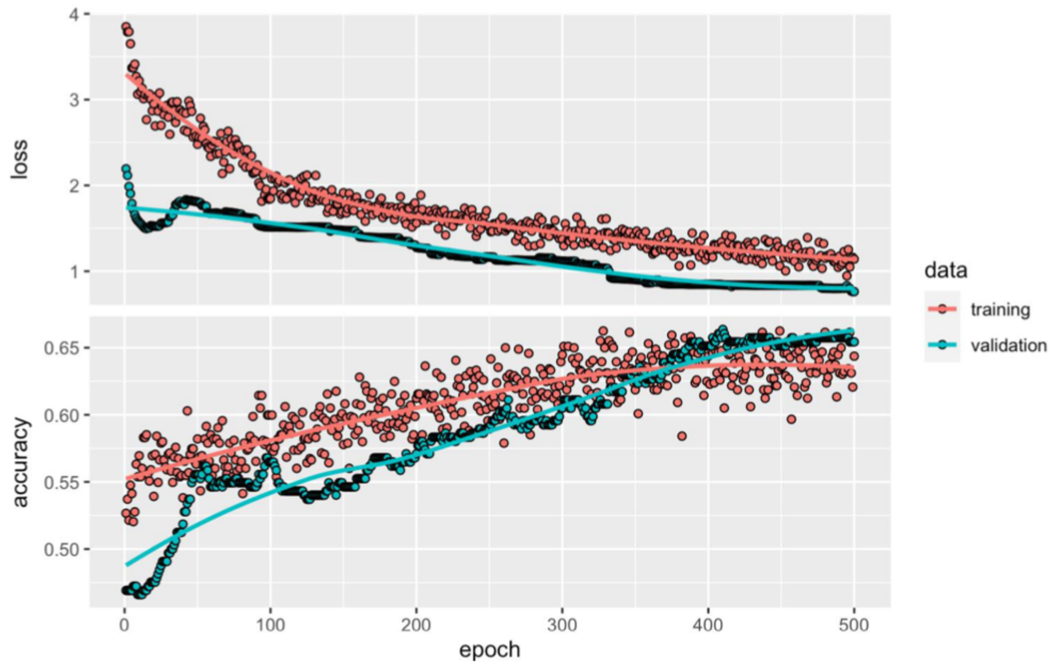


Figure 10: History graph

The Figure 10 describes the variable history. By default Keras' *model.fit()* returns a History callback object. This object keeps track of the accuracy, loss, and other training metrics, for each epoch, in the memory. The epoch is settled to 500 and the batch_size to 256. We can observe a decrease in the loss for the train data set and validation one, and an increment in accuracy also in both data sets.

Next, computation of the contingency matrix of the train data set.

	low	***high***	***Sum***
low	445	15	460
high	0	497	497
Sum	445	512	957

Table 7: Contingency matrix on train data set

By have a look at Table 7 in the train data set with the neural network there is an accuracy rate of 98.43%, and an error rate of 1.57%.

	low	***high***	***Sum***
low	151	3	154
high	0	170	170
Sum	151	173	324

Table 8: Contingency matrix on validation data set

Instead, in the contingency matrix based on the validation data set, Table 8, there is an accuracy rate of 99.08% and an error rate of 0.92%.

4 Result comparison

The goal of this chapter is to evaluate the models used to train the validation data set. I based my choice on the accuracy, a metric that summarizes the performance of the models used (tot. number of correct predictions / tots. number of predictions).

Which is the model with the highest accuracy rate?

Training model	***Accuracy rate***
Logistic regression	74.07%
Random forest	81.80%
Neural network	99.08%

Table 9: Models' accuracy

Based on the accuracy rate find out, I can attest that the neural network model is the best one, with an accuracy above the 99%. This means that neural network seems to be the best model to predict the target variable quality.

Application of the neural network on the test data set with the predict function. By applying the method, we can see that there are 103 values that belongs to low quality, and the other 215 that belongs to high quality. Please note that this results are carried out by using the *set_random_seed()*, a function that establish a common random seed for R, Python and TensorFlow.

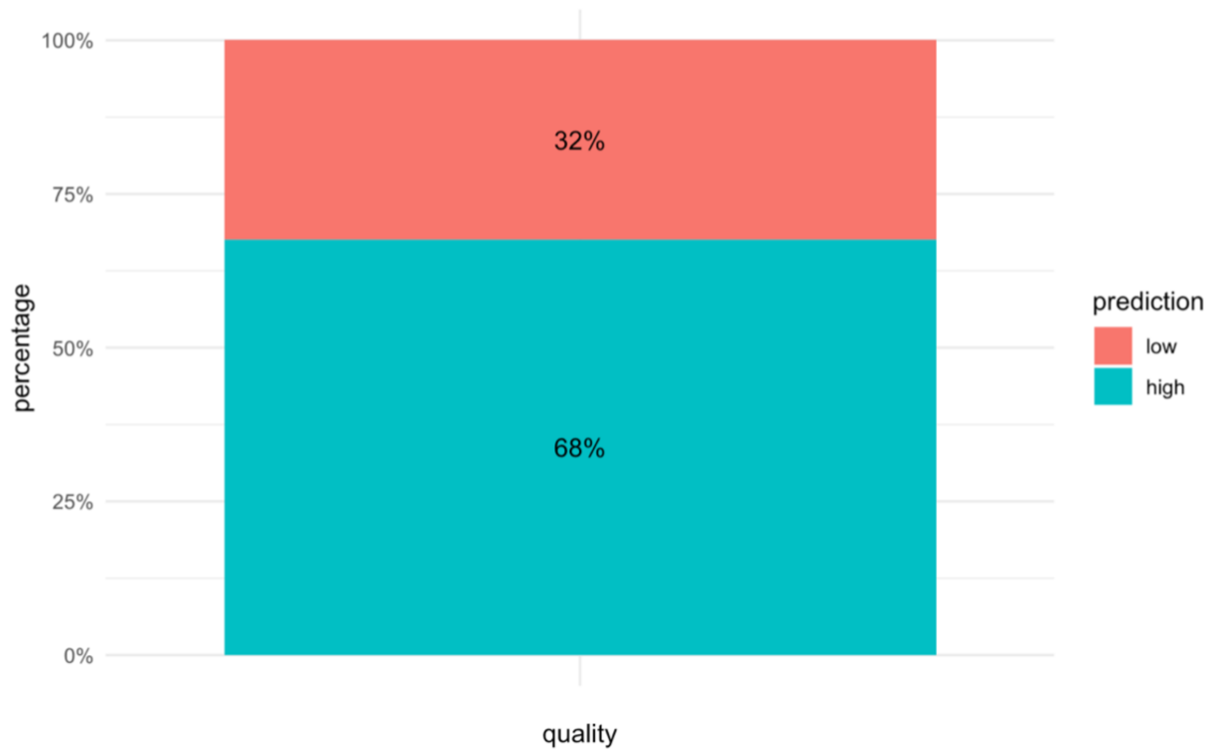


Figure 11: Bar plot of predicted values in test data set

The neural network prediction are displayed in Figure 11, that describes the presence of the target variable quality in the test data set. It can be said by have a look to the results, that the predicted value of quality in the test data set are high quality wine for the 68% and low quality wine for the 32% of the observations of the test data set.

5 Conclusion

- Which are the variables that present the highest correlation coefficient?

Based on the multivariate scatter plot of Figure 6, the variables mostly correlated are *citric.acid* and *fixed.acidity*.

- Which are the variables that present the lowest correlation coefficient?

Based on the multivariate scatter plot of Figure 6, the variables with the lowest correlation coefficient are *pH* and *fixed.acidity*.

- Which variables contribute the most to target variable **quality**?

According to Figure 9, the variables that contribute the most are, at first *alcohol*,at second *sulphates*.

- How can be predicted the target variable **quality**?

Based on this analysis, I used 3 possible different approaches to predict the variable *quality*; according to the results the best method is the neural network. I'm going to add another question, that may clarify some doubts.

- Is the neural network a good model to predict the values on the test data set?

Well, there are several possible answers to this question. I think that, based on the accuracy there is no doubt about the neural network's power; I found some difficult to carry out the predicted values of my neural network based on test data set cause the predicted values always changed when I did the computation of the network structure and prediction.

I think this problem is related to the continue initialization of the neural network and also cause the *Keras()* package is compiled in Python. In order to avoid different types of predicted values, I settled the a *set_random_seed()*, that helped me to obtain the same history and predicted values at every computation.

6 Appendix

```
#libraries used
library(ggplot2)
library(scales)
library(skimr)
library(visdat)
library(viridis)
library(keras)
library(randomForest)
library(bestglm)
library(tab)
library(dplyr)
library(mmand)
library(GGally)
library(corrplot)
library(DataExplorer)
library
#dataset load and its structure
dataset<-read.csv("winequality-red_train.csv") dataset$X=NULL
as.factor(dataset$quality) table(dataset$quality)
#eda,univariate analysis vis_dat(dataset)
skim(dataset) plot_histogram(dataset-dataset$quality)
plot_density(dataset-dataset$quality)
plot_qq(dataset-dataset$quality)
#eda,multivariate analysis
R<-cor(dataset[1:11])
print(htmlTable::htmlTable(R,useViewer=TRUE))
corrplot.mixed(R, upper="color", tl.col="black",tl.cex=0.8)
ggpairs(dataset, upper=list(continuous=wrap("cor",size=2)),
    lower=list(continuous=wrap("points",alpha=0.2,size=0.1),combo=
wrap("dot",aplha=1,size=0.2)))
plot_boxplot(dataset, by="quality")
plot_qq(dataset, by="quality")
#preparing data for logistic regression
dataset.levels<-dataset
dataset.levels[dataset.levels$quality<6,'quality']<-"low"
dataset.levels[dataset.levels$quality!="low",'quality']<-"high"
dataset.levels$quality<-factor(dataset.levels$quality,levels=c("lo
w","high"))
table(dataset.levels$quality)
#logistic regression
glmfit1=glm(dataset.levels$quality ~.,family=binomial,data=dataset
.levels)
#summary(glmfit1, digits=3)
tabglm(glmfit1,columns =c("beta.se","test","p"))
glmfit1=update(glm(dataset.levels$quality~volatile.acidity+
)
chlorides+free.sulfur.dioxide+
total.sulfur.dioxide+sulphates+
alcohol, family=binomial,data=dataset.levels)
tabglm(glmfit1,columns =c("beta.se","test","p"))
glm.probs=predict.glm(glmfit1, type="response")
L<-nrow(dataset.levels)
glm.pred <- rep(\low", L)
glm.pred[glm.probs>.5]="high"
table(glm.pred,dataset.levels$quality)
confMat<-addmargins(table(glm.pred,dataset.levels$quality))
confMat
accuracy1=(confMat[1,1]+confMat[2,2])/L*100
err1=100-accuracy1
err1
accuracy1
#validation dataset prep
datasetvalid<-read.csv("winequality-red_validation.csv")
datasetvalid$X=NULL
datasetvalid[datasetvalid$quality<6,"quality"]<-"low"
datasetvalid[datasetvalid$quality!="low","quality"]<-"high"
datasetvalid$quality<-factor(datasetvalid$quality,levels=c("low","
high"))
table(datasetvalid$quality)
#start the work for the confusion matrix
glm.probs2=predict.glm(glmfit1,newdata=datasetvalid,type="response
")
Lv<-nrow(datasetvalid)
glm.predvalid<-rep("low", Lv)
glm.predvalid[glm.probs2>.5]="high"
confmatvalid<-addmargins(table(glm.predvalid,datasetvalid$quality)
)
confmatvalid
accuracy2=(confmatvalid[1,1]+confmatvalid[2,2])/Lv*100
err2=100-accuracy2
accuracy2
err2
#application of random forest
bag.boston=randomForest(factor(dataset.levels$quality)~.,dataset.l
evels,ntree=500,mtry=6)
bag.boston
#output of the tree number 1
getTree(bag.boston,1)
#variable importance
importance=importance(bag.boston)
variable_importance=data.frame(Variables = row.names(importance),
Importance = round(importance[ ,'MeanDecreaseGini'],2))
#create a rank variable based on importance
rankImportance=variable_importance %>%
mutate(Rank = paste0('',dense_rank(desc(Importance))))
ggplot(rankImportance, aes(x = reorder(Variables, Importance),
y = Importance, fill = Importance)) +
geom_bar(stat='identity') +
geom_text(aes(x = Variables, y = 0.5, label = Rank),
hjust=0, vjust=0.55, size = 7, colour = 'red') +
labs(x = 'Variables') +
coord_flip() +
theme_grey()
#data validation
N=nrow(dataset.levels)
random.probs=predict(bag.boston,data=dataset.levels,type="class")
confMatRandomForest=addmargins(table(dataset.levels$quality,random
.probs))
print(htmlTable::htmlTable(confMatRandomForest,useViewer=TRUE))
accuracy3=(confMatRandomForest[1,1]+confMatRandomForest[2,2])/N*10
0
err3=100-accuracy3
```

```
accuracy3
err3
#data validation on validation dataset
NV=nrow(datasetvalid)
random.probsValidation=predict(bag.boston,newdata=datasetvalid,typ
e="class")
confMatRandomForest=addmargins(table(datasetvalid$quality,random.p
robsValidation))
print(htmlTable::htmlTable(confMatRandomForest,useViewer=TRUE))
accuracy4=(confMatRandomForest[1,1]+confMatRandomForest[2,2])/NV*1
00
err4=100-accuracy4
accuracy4
err4
#neural networks model
#data preparation x=as.matrix(scale(dataset.levels[1:11]))
y=dataset.levels$quality
#model with 3 blocks, with 7 units Tensorflow::set_random_seed(42)
modnn=keras_model_sequential()
modnn %>%
layer_dense(units=7,activation="relu",input_shape=c(11))%>%
layer_dropout(0.3) %>%
#this means dropout of 30%,the 30% of neurons pass to the next layer
layer_batch_normalization()%>% #normalize the values within the batch
  layer_dense(units=7,activation="relu")%>%
  layer_dropout(0.3)%>%
  layer_batch_normalization()%>%
  layer_dense(units=7,activation="relu")%>%
  layer_dropout(0.3)%>%
  layer_batch_normalization()%>%
  layer_dense(units=1)
#5 number of neurons, units[(n.ofinput+n.ofoutput) / 2] +1
modnn %>% compile(loss="binary_crossentropy",optimizer="adam",metr
ics=c("accuracy"))
summary(modnn)
#compare accuracy of train and validation dataset for neural netwo
rks results
xValidation=as.matrix(scale(datasetvalid[1:11]))
```

```
yValidation=datasetvalid$quality
history=modnn %>% fit(
x,y,validation_data=list(xValidation,yValidation),epochs=500,ba
tch_size=256, verbose=0)
#confusion matrix neural network on train dataset
npred=rep(0, N)
npred[modnn %>% predict(x) > threshold(modnn %>% predict(x), datas
et.levels$quality)]=1
confMatNN=addmargins(table(npred,dataset.levels$quality))
confMatNN
#print(htmlTable::htmlTable(confMatNN,useViewer=TRUE))
accuracy5=(confMatNN[1,1]+confMatNN[2,2])/N*100
accuracy5
err5=100-accuracy5
err5
#confusion matrix on validation dataset
npred2=rep(0,NV)
npred2[modnn %>% predict(xValidation) > threshold(modnn %>%
predict(xValidation),datasetvalid$quality)]=1
confMatNNV=addmargins(table(npred2,datasetvalid$quality))
confMatNNV
#print(htmlTable::htmlTable(confMatNNV,useViewer=TRUE))
accuracy6=(confMatNNV[1,1]+confMatNNV[2,2])/NV*100
accuracy6
err6=100-accuracy6
err6
#application of the best model to test dataset, NEURAL NETWORK
#test dataset load and preparation
dataset_test<-read.csv("winequality-red_test.csv")
dataset_test$X=NULL
xtest=as.matrix(scale(dataset_test[1:11]))
#prediction
dataset_test$prediction=predict(object=modnn,x=xtest,type="respons
e")
dataset_test[dataset_test$prediction<0.5,'prediction']<-"low"
dataset_test[dataset_test$prediction!="low",'prediction']<-"high"
dataset_test$prediction<-factor(dataset_test$prediction,levels=c("
low","high"))
```

```r
table(dataset_test$prediction)
#plotting the predicted values
ggplot(dataset_test,aes(x="",fill=prediction))+
geom_bar(position=position_fill())+
geom_test(aes(label=paste0(percent(..count../sum(..count..)))),
stat="count",position=position_fill(vjust=0.5))+
  scale_y_continuous(labels=percent)+
  xlab("quality")+
  ylab("percentage")+
  theme_minimal()
#create a file with the predicted values
write.csv(dataset_test, 'dataset_test_prediction.csv')
```